Descriptif général du projet

Jamila Sam & Jean-Cédric Chappelier, 2018

Version : 1.0

Introduction

Réaliser un projet de programmation complet, par opposition au codage de simples exercices, est indispensable pour bien ancrer les notions acquises. Vous aurez donc dans le cadre de ce MOOC complémentaire à mettre en œuvre un tel projet avec pour objectif de consolider vos connaissances en programmation et de vous faire pratiquer intensivement les concepts essentiels de la programmation orientée-objet : abstraction et encapsulation, héritage, méthodes virtuelles et polymorphisme.

Description

Le choix du thème du projet s'est porté sur un sujet suffisamment général pour intéresser un large public. Il s'agit de la simulation de colonies d'insectes sociaux et plus précisément celui de fourmilières, tel qu'illustré en figure 1 page suivante. La notion d'*intelligence collective* est sans doute l'une des raisons à l'origine de l'intérêt souvent porté à cette thématique : chaque membre de la colonie n'obéit individuellement qu'à des règles comportementales très simples, mais il émerge de ces comportements simples une auto-organisation complexe et « intelligente » de la colonie, permettant typiquement la survie de la fourmilière dans notre cas.

Le but du projet est donc de créer une application simulant des colonies de fourmis en quête de nourriture. La figure 1 donne un exemple d'état possible de la simulation à produire.

Le modèle simple qui vous est proposé est constitué :

- d'un environnement ;
- de fourmis ;
- et de prédateurs.

Les fourmis et l'environnement interagissent :

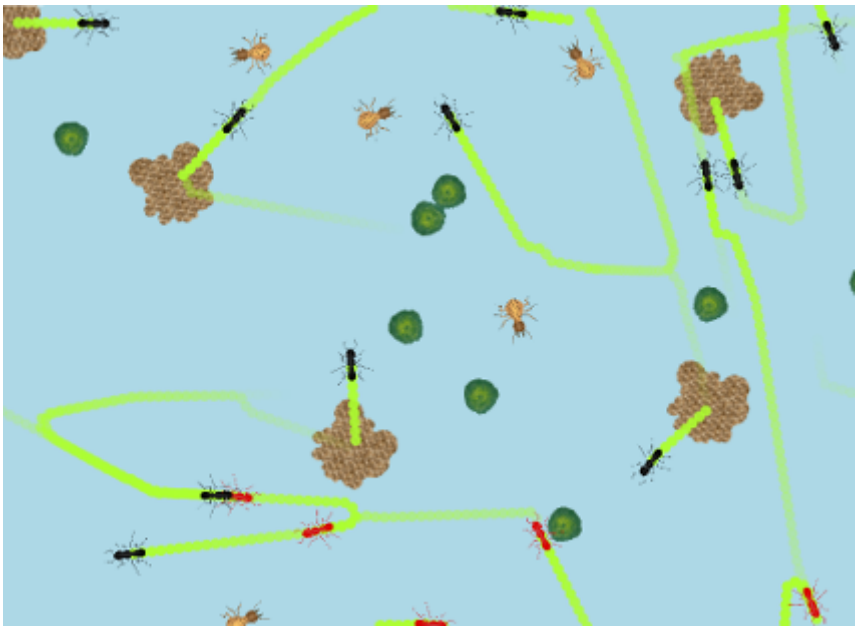

Figure 1: Simulation de fourmilières

- l'environnement contient les fourmilières auxquelles appartiennent les fourmis ;
  de plus, il génère périodiquement des sources de nourriture aléatoirement dispersées dans lesquelles les fourmis peuvent aller puiser ;
- les fourmis collectent la nourriture pour la ramener à leur fourmilière et déposent dans l'environnement des traces de phéromone qui seront utilisées par leur congénères pour se déplacer ; ces traces vont leur permettre notamment de retrouver leur fourmilière plus facilement ou de privilégier les chemins allant vers de la nourriture ; la phéromone déposée s'évapore au cours du temps et peut finir par disparaître.

Deux types de fourmis seront simulées : les ouvrières et les soldates. Les premières collectent la nourriture alors que les secondes ont pour rôle de défendre le territoire contre des prédateurs. Les prédateurs seront soit des termites, soit des fourmis de fourmilières concurrentes. Ils ont pour rôle principal de réguler les populations de fourmis dans la simulation.

Deux modes de déplacement sont possibles pour les animaux simulés (fourmis ou termites) :

- le déplacement inertiel : les animaux se déplacent au hasard en privilégiant les directions qui leur font face ; c'est ce mode de déplacement qu'utiliseront les termites ainsi que les fourmis en l'absence de phéromone ;

- le déplacement sensoriel : utilisé par les fourmis pour suivre les traces de phéromone déposées dans l'environnement.

L'environnement est modélisé comme un terrain rectangulaire en deux dimensions. Il est cependant géré de façon *torique* : les animaux dépassant les limites de l'environnement par un côté réapparaîtront sur le côté opposé (cela sera expliqué plus en détail le moment venu).

A titre d'exemple, on peut voir sur la figure 1 un environnement contenant des fourmilières (petits tas de terre), des fourmis ouvrières (en noir), soldates (en rouge) et des termites (brunes bicolores). Les fourmis laissent sur le sol des traces de phéromone (en vert) qui s'évaporent au cours du temps (elles s'éclaircissent). Les petits tas vert foncé représentent de la nourriture dans laquelle les fourmis peuvent aller puiser.

Premiers pas vers la conception d'un programme

On peut se douter que chacun des éléments décrits ci-dessus (environnement, fourmi, termite, nourriture, phéromone, fourmilière etc.) aura un équivalent Java dans le code que vous allez produire. En concevant le programme, il faut non seulement recenser les éléments à modéliser, mais aussi se poser un certain nombre de questions comme :

- comment les éléments intervenant dans la simulation interagissent-ils entre eux ?
- ont-ils des points communs, ou sont-ils totalement indépendants ?

Au vu de la description qui précède, les classes suivantes semblent s'imposer d'emblée :

- une classe pour représenter l'environnement à simuler : ce dernier recensera les animaux présents, les sources de nourriture disponibles, la phéromone déposée, etc. ;
- des autres pour les animaux, bien sûr, sans lesquels rien ne se passerait ;
- une classe pour représenter les sources de nourriture ;
- une classe pour les fourmilières.

Plus finement, il apparaîtra rapidement qu'il y a des éléments communs à tous les animaux comme, par exemple, le fait qu'ils aient une position dans l'environnement et une direction de déplacement. Il existera aussi des différences, au niveau du comportement par exemple, les ouvrières ayant un rôle différent de celui des soldates.

On peut par exemple imaginer une classe de base (`Animal`) pour spécifier les caractéristiques et méthodes communes à tous les animaux du monde. Il faudra

ensuite *spécialiser* cette classe de base (notion d'héritage) pour caractériser les attributs et méthodes spécifiques aux fourmis et aux termites.

De façons similaire, on peut ainsi anticiper les différents éléments intervenant dans la simulation et proposer progressivement une conception possible pour le programme. Vous trouverez au fur et à mesure que vous progresserez dans le projet, des schémas vous expliquant l'architecture du programme à laquelle nous vous proposons d'adhérer.

Il est indispensable de la comprendre pour pouvoir l'implémenter correctement. Il est évident que d'autres solutions plus simples, plus compliquées, ou simplement différentes existent et chacune (simple ou compliquée) a généralement des avantages et des inconvénients. Toutefois, la modélisation que nous vous proposons est raisonnable, et la suivre sera indispensable pour bénéficier de la correction automatique et obtenir les points attribués à chaque étape.

Technique générale de simulation

Une façon simple de simuler l'évolution de l'environnement consiste à invoquer en boucle la mise à jour de ses constituants qui évoluent dans le temps. Chaque itération correspondra à un *cycle de simulation*.

La classe représentant l'environnement comportera donc une méthode de mise à jour dont le rôle sera de parcourir l'ensemble des entités présentes pour gérer les actions nécessaires à chaque cycle. Pour les animaux, par exemple, il s'agira du déplacement, des combats ou du transport de nourriture. Pour les phéromones, ce sera l'évaporation, etc.

Une description plus détaillée des éléments à simuler et des techniques employées sera bien sûr fournie dans les énoncés des différentes étapes du projet.

Interface graphique

Une interface graphique est fournie pour vous permettre de visualiser vos progrès, en observant l'évolution de l'environnement. La bibliothèque Java contient trois ensembles de classes permettant la création d'interfaces utilisateurs graphiques (graphical user interfaces ou GUIs). JavaFX est celle qui est utilisée dans le matériel que nous vous fournissons.

Par contre, l'étude détaillle de JavaFX sort de la portée de ce MOOC. Seuls les éléments indispensables à son utilisation dans le cadre de ce projet seront expliqués dans l'énoncé lorsque nécessaire.

Etapes du projet

Le projet est séparé en 14 étapes, chaque étape étant une extension de la précédente : c'est-à-dire qu'elles sont dépendantes l'une de l'autre et que vous devez achever une étape avant d'entamer la suivante. La figure 2 présente une vue générale du déroulement du projet. Des classes sont ajoutées en cours de route (notée par un +), d'autres sont achevées (en vert) et beaucoup d'entre elles sont modifiées chaque semaine (les classes modifiées dans une étape sont listées sous celle-ci). L'énoncé détaillé du projet vous donnera, étape par étape, les instructions nécessaires à leur réalisation.

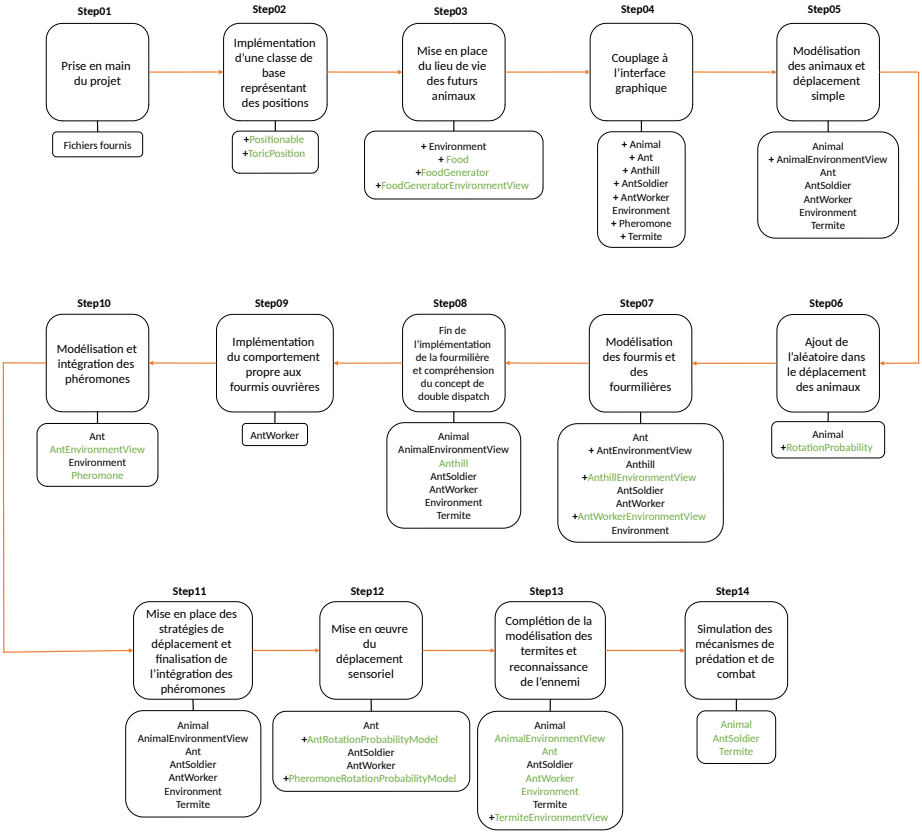

Figure 2: Les 14 étapes du projet.

Références

- https://interstices.info/lintelligence-en-essaim-ou-comment-faire-complexe-avec-du-simple/

- https://openclassrooms.com/fr/courses/1469021-les-applications-web-avec-javafx/1469344-presentation-de-linterface-graphique-en-javafx